```
                                 ****************************************
                                 *THIS PROGRAM IS IN THE EDITOR/ASSEMBLER  *
                                 *ON PAGE 347, THE AUTOMATIC SPRITE MOTION *
                                 *EXAMPLE.  THIS IS THE MINI-MEMORY VERSION.*
                                 ****************************************

CO DATA >FF00
BA DATA >3C7E,>FFFF,>FFFF,>7E3C
SD DATA >6178,>8006
   DATA >6178,>8003
   DATA >6178,>8004
   DATA >6178,>800B,>D000
SP DATA >0404,>0000,>F808,>0000
   DATA >0CF4,>0000,>F0F0,>0000
MY BSS  >20
MO LWPI MY                    *LOAD REGISTERS
*
   LI   R0,>384
   MOVB @CO,R1
   BLWP @>6024                *LOAD BACKGROUND COLOR AS WHITE
*
   LI   R0,>400
   LI   R1,BA
   LI   R2,8
   BLWP @>6028                *LOAD BALL PATTERN
*
   LI   R0,>300
   LI   R1,SD
   LI   R2,17
   BLWP @>6028                *LOAD SPRITE ATTRIBUTE LIST
*
   LI   R0,>780
   LI   R1,SP
   LI   R2,16
   BLWP @>6028                *LOAD SPEED OF SPRITES
*
   LI   R0,>81E1
   BLWP @>6034                *LOAD VDP REGISTER TO MAGNIFY SPRITES
*
   LI   R1,4
   SLA  R1,8
   MOVB R1,@>837A             *SPECIFY NUMBER OF SPRITES
*
LO LI   R0,>300
   LI   R3,4                  *REPEAT 4 TIMES
   LI   R2,2
*
L2 LIMI 2                     *ENABLE INTERRUPT
   LIMI 0                     *DISABLE INTERRUPT
   LI   R1,MY+14              *READ IT INTO REGISTER 7
   BLWP @>6030
*
   AI   R7,-24
   CI   R7,>B8C8              *CHECK IF 0<Y<184,24<X<224
   JH   AD
*
NE AI   R0,4                  *LOOK AT NEXT SPRITE
   DEC  R3
   JEQ  LO
   JMP  L2
```

```
*************************************************************
*                                                           *
* THE FOLLOWING ASSEMBLY PROGRAM DEMONASTRATES HOW TO:      *
*       1) TAKE A STRING FROM TI BASIC                      *
*       2) ADD >60 TO EACH CHARACTER                        *
*       3) PRINT THE STRING TO THE SCREEN                   *
*       4) RETURN TO TI BASIC                               *
*                                                           *
*************************************************************
       DEF  START          DEFINE ENTRY POINT OF PROG.
       REF  VSBW,STRREF,GPLWS
STATUS EQU  >837C
SAVE$  BSS  >FF            STRING BUFFER: 255 BYTES.
MYWKSP BSS  >20
SAVRT  DATA 0
FULL$  BYTE >FF
*
START  MOV  R11,@SAVRT     SAVE ENTRY ADDRESS.
       LWPI MYWKSP         LOAD MY WORKSPACE.
*
       CLR  R0
       LI   R2,SAVE$       ADDRESS OF STRING BUFFER.
CLEAR$ MOVB R0,*R2+        CLEAR AT STRING BUFFER.
       CI   R2,SAVE$+>FE END OF STRING BUFFER.
       JNE  CLEAR$         NO. GO BACK TO CLEAR$.
       MOVB @FULL$,@SAVE$
*
       LI   R1,1           SET UP TO GET STRING FROM BASIC.
       LI   R2,SAVE$       ADDRESS OF STRING BUFFER.
       BLWP @STRREF        GET STRING.
*
       LI   R4,SAVE$       ADDRESS OF STRING BUFFER.
       LI   R3,>6000       HEX 60 (LEFT JUSTIFIED).
       LI   R2,>0100       HEX 01 (LEFT JUSTIFIED).
       LI   R1,SAVE$       ADDRESS OF STRING BUFFER.
CONVRT INC  R1
       AB   R3,*R1
       CB   R2,*R4         END OF STRING?
       JEQ  WRITE          YES, WRITE IT TO SCREEN.
       AI   R2,>0100       INCREMENT LEFT BYTE IN R2.
       JMP  CONVRT
*
WRITE  LI   R0,390         VDP SCREEN ADDRESS.
       LI   R2,>0100       HEX 01 (LEFT JUSTIFIED).
       LI   R3,SAVE$+1     1st CHARACTER OF STRING.
*
WRITE1 MOVB *R3+,R1        MOVE CHARACTER AT R3 ADDRESS TO
*                          R1 AND INCREMENT R3.
       BLWP @VSBW
       CB   R2,*R4         END OF STRING?
       JEQ  GOBACK         YES, GO BACK TO BASIC.
       INC  R0             SCREEN ADDRESS BY 1.
       AI   R2,>0100       INCREMENT LEFT BYTE IN R2.
       JMP  WRITE1
*
GOBACK CLR  R0
       MOVB R0,@STATUS     CLEAR STATUS BYTE.
       LWPI GPLWS          SEE p.442 E/A MANUAL.
       B    @>0070         GO BACK TO BASIC.
*
       END
```

KSCAN FOR MINI-MEMORY

```
                         ************************************************
                         *THIS PROGRAM GIVES AN EXAMPLE OF AN MINI-MEMORY   *
                         *SUBROUTINE.  WHEN THE SPACE BAR IS PRESSED CONTROL*
                         *BRANCHES BACK TO THE MAIN ROUTINE.                *
                         ************************************************

KS EQU  >6020            *LOCATION OF KSCAN ROUTINE
ST EQU  >837C            *>837C IS THE STARTING LOCATION OF THE STATUS BYTE
KY EQU  >8375            *ASCII VALUE OF KEY PRESSED,RETURN HERE
SE DATA >2000            *IF KEY IS PRESSED STATUS WILL BE=TO THIS

   *SET UP KEYBOARD FOR INPUT
   *WHEN KSCAN IS CALLED BIT 2 OF THE STATUS BYTE IS SET IF KEY IS PRESSED

TE CLR  @KY              *ENTRY POINT OF PROGRAM,CLEAR KEYBOARD
   BLWP @KS              *SET KEY SCAN SUBROUTINE
   MOV  @ST,R3           *MOVE STATUS BYTE FOR COMPARISON
   COC  @SE,R3           *COMPARE SET WITH STATUS BYTE
   JEQ  KT               *IF KEY HIT JUMP TO BACK
   JMP  TE               *IF KEY HASN'T BEEN HIT JUMP TO TEST

   *WHEN A KEY IS TOUCHED REGISTERS MUST BE REINITIALIZED
   *IN PREPARATION FOR RETURNING TO CALLING ROUTINE
   *TO RETURN TO GPL THE STATUS BYTE MUST BE CLEARED

KT LI   RO,>2000         *LOAD ASCII HEX CODE FOR SPACE BAR INTO
                         *REGISTER RO.  KT="KTEST"
   CB   RO,@KY           *TEST IF SPACE BAR WAS PRESSED
   JEQ  BK               *YES,JUMP TO RETURN ROUTINE
   JMP  TE               *NO,JUMP TO KEYBOARD TEST ROUTINE

   *RETURN ROUTINE
BK CLR  RO               *RETURN TO MENU SCREEN
   MOVB RO,@ST           *CLEAR STATUS BYTE
   LWPI >83E0
   B    @>0070
   AORG >701C
   DATA >7F04
   DATA >7FE0
   AORG >7FE0            *GETS TO THE NEW ENTRY POINT IN THE TABLE
   TEXT 'TESTIT'         *ENTERS THE PROGRAM NAME TESTIT INTO THE
                         *STORED BYTES BEGINNING AT LOCATION >7FE0
   DATA TE               *THE LABEL TE IS EQUATED TO THE ADDRESS
                         *WHICH IS THE ENTRY POINT IN THE PROGRAM.
```

```
* THIS ROUTINE ALLOWS "SIMULTANEOUS" INPUT FROM KEYBOARDS 1
* AND 2 OR JOYSTICKS 1 AND 2. ALL FOUR SQUARES MAY BE USED
* AT THE SAME TIME. THE PROGRAM NAME IS "KEYSCN".

        DEF   KEYSCN
        REF   VSBW,VSBR,VMBW,GPLWS,SCAN,VMBR
MYWS    EQU   >8300           LOCATION OF PROGRAM WORKSPACE
CPURAM  EQU   >8300           STARTING ADDRESS, CPU RAM
KEYBRD  EQU   CPURAM+>74      LOCATION OF KEYBOARD NUMBER
KEY     EQU   CPURAM+>75      RETURN ASCII VALUE OF KEY PRESSED
JOYY    EQU   CPURAM+>76
JOYX    EQU   CPURAM+>77
SAL     EQU   >0300           STARTING ADDRESS, SPRITE ATTRIBUTE LIST
SDT     EQU   >0400           STARTING ADDRESS, SPRITE DESC. TABLE
*
H00     BYTE >00              X,M KEYS & BOUNDARY
H01     BYTE >01              INC, DEC VALUE
H02     BYTE >02              S,J KEYS
H03     BYTE >03              D,F KEYS
H05     BYTE >05              E,I KEYS
H0D     BYTE >0D              Q,Y KEYS
H10     BYTE >10              BOUNDARY
H12     BYTE >12              V,. KEYS
HE7     BYTE >E8              BOUNDARY
HFF     BYTE >FF              NEGATIVE 1
HB7     BYTE >B7              BOUNDARY
*
*DEFINE FOUR SPRITES USING PATTERN >80
H0000   DATA >0000
SPRPAT  DATA >FF81,>8181,>8181,>81FF          SPRITE PATTERN
SALINI  DATA >0080,>8001,>B780,>8001          SPRITE INITIAL ATTRIBUTES
        DATA >6010,>8001,>60E8,>8001
        DATA >D000
*
KEYSCN  LIMI 0                DISABLE INTERRUPTS
        LWPI MYWS             USE PROGRAM WORKSPACE
        LI   R0,SDT
        LI   R1,SPRPAT        GET SPRITE PATTERN
        LI   R2,8             PREPARE 8 BYTES TO MOVE TO VDP
        BLWP @VMBW
        LI   R0,SAL
        LI   R1,SALINI        SPRITES' INITIAL ATTRIBUTES
        LI   R2,17            PREPARE 17 BYTES TO MOVE TO VDP
        BLWP @VMBW
******************
*MAIN PROGRAM LOOP*
******************
LOOP    BL   @INPUT           GET INPUT
        JMP  LOOP
*************************************
*THIS IS THE RTN TO GET SOME FORM OF  *
*INPUT, EITHER FROM KEYBRD OR JOYSTICK*
*************************************
INPUT   MOV  R11,R14          SAVE RETURN LINKAGE
        CLR  @KEYBRD          INITIALIZATION TO KEYBOARD 0
BRDINC  AB   @H01,@KEYBRD     INCREMENT
        CB   @KEYBRD,@H03     TEST FOR KEYBOARD 3
        JEQ  INPTRT           YES, RETURN
        BL   @GETINP          GET INPUT
        CB   @KEY,@HFF        HAS A KEY BEEN PRESSED?
```

```
       JEQ  CHKJOY          NO, CHECK JOYSTICKS
********************************
*DETERMINE WHAT KEY WAS PRESSED *
********************************
CHKKEY CLR  R3              CLEAR R3
       MOVB @KEY,R3         MOVE INPUT VALUE TO R3
WATKEY CB   R3,@H05         UP KEY?
       JEQ  UP
       CB   R3,@H00         DOWN KEY?
       JEQ  DOWN
       CB   R3,@H02         LEFT KEY?
       JEQ  LEFT
       CB   R3,@H03         RIGHT KEY?
       JEQ  RIGHT
       CB   R3,@H0D         FIRE?
       JEQ  FIRE
       CB   R3,@H12         FIRE?
       JEQ  FIRE
CHKJOY C    @JOYY,@H0000 CENTER?
       JEQ  BRDINC          YES, SCAN NEXT KEYBOARD
       CB   @JOYY,@H00
       JGT  UP
       JLT  DOWN
       CB   @JOYX,@H00
       JGT  RIGHT
       JLT  LEFT
**************************************
*JUMP MAY ONLY BE WITHIN >100 BYTES   *
*OF THE INSTRUCTION,THEREFORE THESE   *
*NEXT LINE ALLOW THOSE CALLED SUB-    *
*ROUTINES TO BE PLACED ANYWHERE IN    *
*THE PROGRAM                          *
**************************************
UP     B    @UPS            GOTO "UP" SUBROUTINE
DOWN   B    @DOWNS          GOTO "DOWN" SUBROUTINE
LEFT   B    @LEFTS          GOTO "LEFT" SUBROUTINE
RIGHT  B    @RIGHTS         GOTO "RIGHT" SUBROUTINE
FIRE   B    @BRDINC         INSERT USER DEFINED FIRE SUB. HERE
*************************************
*THIS NEXT LABEL IS THE RETURN POINT *
*TO BE USED BY ALL THE CALLED SUB-   *
*ROUTINES. THIS RETURNS CONTROL TO   *
*WHOMEVER REQUESTED THE INPUT WITH A *
*BL @ INPUT STATEMENT                *
*************************************
INPTRT MOV  R14,R11         RESTORE RETURN LINKAGE
       B    *R11
***********************************
*THIS ROUTINE WILL MOVE SPRITES UP *
*AT A RATE OF ONE PIXEL            *
***********************************
UPS    LIMI 0               DISABLE INTERRUPTS
       CLR  R1              PREPARE FOR VSBR
       LI   R0,SAL+8        GET SPRITE Y POSITION
       BLWP @VSBR
       CB   R1,@H00         BOUNDARY CHECK
       JEQ  UPRT
       SB   @H01,R1         DECREMENT Y
       BLWP @VSBW
       LI   R0,SAL+12       MOVE PARALLEL SPRITE
```

```
       BLWP @VSBW
       LIMI 2            ENABLE INTERRUPTS
UPRT   B    @BRDINC      CHECK NEXT KEYBRD OR JOYST
************************************
*THIS ROUTINE WILL MOVE SPRITES DOWN*
*AT A RATE OF ONE PIXEL             *
************************************
DOWNS  LIMI 0            DISABLE INTERRUPTS
       CLR  R1           PREPARE FOR VSBR
       LI   R0,SAL+8     GET Y OR SPRITE
       BLWP @VSBR
       CB   R1,@HB7      BOUNDARY CHECK
       JEQ  DOWNRT
       AB   @H01,R1      INCREMENT Y
       BLWP @VSBW
       LI   R0,SAL+12    MOVE PARALLEL SPRITE
       BLWP @VSBW
       LIMI 2            ENABLE INTERRUPTS
DOWNRT B    @BRDINC      CHECK NEXT KEYBRD OR JOYST
************************************
*THIS ROUTINE WILL MOVE SPRITES LEFT *
*AT A RATE OF ONE PIXEL             *
************************************
LEFTS  LIMI 0            DISABLE INTERRUPTS
       CLR  R1           PREPARE FOR VSBR
       LI   R0,SAL+1     GET X OF SPRITE
       BLWP @VSBR
       CB   R1,@H10      BOUNDARY CHECK
       JEQ  LEFTRT
       SB   @H01,R1      DECREMENT X
       BLWP @VSBW
       LI   R0,SAL+5     MOVE PARALLEL SPRITE
       BLWP @VSBW
       LIMI 2            ENABLE INTERRUPTS
LEFTRT B    @BRDINC      CHECK NEXT KEYBRD OR JOYST
************************************
*THIS ROUTINE WILL MOVE SPRITES RIGHT*
*AT A RATE OF ONE PIXEL             *
************************************
RIGHTS LIMI 0            DISABLE INTERRUPTS
       CLR  R1           PREPARE FOR VSBR
       LI   R0,SAL+1     GET X OF SPRITE
       BLWP @VSBR
       CB   R1,@HE7      BOUNDARY CHECK
       JEQ  RIGHRT
       AB   @H01,R1      INCREMENT X
       BLWP @VSBW
       LI   R0,SAL+5     MOVE PARALLEL SPRITE
       BLWP @VSBW
       LIMI 2            ENABLE INTERRUPTS
RIGHRT B    @BRDINC      SCAN NEXT KEYBRD AND JOYST
************************************
*STANDARD INPUT SCAN ROUTINE WHICH *
*USES GPL WORKSPACE.               *
*SCANS KEYBOARD AND JOYSTICKS      *
************************************
GETINP MOV  R11,R15
       LIMI 0
       LWPI GPLWS
       BL   @SCAN
```

```
*
* GPLLNK FOR EXTENDED BASIC
* USES ONLY FEW "MAGIC" NUMBER -- THE ADDRESS >2000 WHICH
* IS ASSUMED TO CONTAIN THE ENTRY ADDRESS FOR THE XML IN CALL LINK
* UTILWS AT >2038
UTILWS EQU >2038
FAC    EQU  >834A
SUBSTK EQU  >8373
GRMRA  EQU  >9802
GRMWA  EQU  >9C02
GPLWS  EQU  >83E0
PAD    EQU  >8300
*
GPLLNK DATA UTILWS,GPLO
*
GPLO   MOVB @GRMRA,R1   FETCH GROM ADDRESS
       SWPB R1
       MOVB @GRMRA,R1
       SWPB R1
       AI   R1,-3         BACK UP TO THE XML INSTRUCTION
       MOVB @SUBSTK,R2    GET STACK POINTER
       SRL R2,8
       AI  R2,PAD
       INCT R2            PUSH XML ADDRESS FOR RETURN
       MOVB R1,*R2
       SWPB R1
       MOVB R1,@1(R2)
       SWPB R2
       MOVB R2,@SUBSTK
       MOVB *R14+,@GRMWA SET UP ADDRESS TO CALL
       MOVB *R14+,@GRMWA SECOND BYTE (ALSO ADJUST RETURN)
       MOV @>2000,R4      SAVE CURRENT XML LINK
       LI  R3,GPL1        NEW XML LINK
       MOV  R3,@>2000
       LWPI GPLWS         GET READY
       RT                 GO TO ROUTINE!
GPL1   LWPI UTILWS        WE SHOULD RETURN HERE
       MOV  R4,@>2000     RESTORE ORIGINAL XML LOCATION
       RTWP               AND GO BACK TO CALLER
*
***
*
*      TEST IT
       DEF TRYGPL
TRYGPL LI  R0,>400
       MOV  R0,@FAC
       BLWP @GPLLNK
       DATA >16           LOAD LARGE CHARACTERS
       RT
       END
```

```
SCLEN  EQU  >8355
SCNAME EQU  >8356
CRULST EQU  >83D0
SADDR  EQU  >83D2
GPLWS  EQU  >83E0          GPL/EXTENDED BASIC WORKSPACE
*
VDPRD  EQU  >8800          VDP read data address
VDPWD  EQU  >8C00          VDP write data address
VDPWA  EQU  >8C02          VDP write address address

FLGPTR DATA 0              Pointer to flag byte in PAB
SVGPRT DATA 0              Save GPL return address
SAVCRU DATA 0              CRU address of peripheral
SAVENT DATA 0              Entry address of DSR
SAVLEN DATA 0              Save device name length
SAVPAB DATA 0              Ptr into device name in PAB
SAVVER DATA 0              Version number of DSR
DLNKWS DATA 0,0,0,0,0
TYPE   DATA 0,0,0,0,0,0,0,0,0,0,0
*
*** Data
*
C100   DATA 100
H20    EQU  $
H2000  DATA >2000
DECMAL TEXT '.'
HAA    BYTE >AA
*
*** Utility BLWP vectors
*
DSRLNK DATA DLNKWS,DLENTR     Link to device service routine
*
*** Link to device service routine
*
DLENTR MOV  *R14+,R5          Fetch program type for link
       SZCB @H20,R15          Reset equal bit
       MOV  @SCNAME,R0        Fetch pointer into PAB
       MOV  R0,R9             Save pointer
       AI   R9,-8             Adjust pointer to flag byte
       BLWP @VSBR             Read device name length
       MOVB R1,R3             Store it elsewhere
       SRL  R3,8              Make it a word value
       SETO R4                Initialize a counter
       LI   R2,NAMBUF         Point to NAMBUF
LNK$LP INC  R0                Point to next char of name
       INC  R4                Increment character counter
       C    R4,R3             End of name?
       JEQ  LNK$LN            Yes
       BLWP @VSBR             Read current character
       MOVB R1,*R2+           Move it to NAMBUF
       CB   R1,@DECMAL        Is it a decimal point?
       JNE  LNK$LP            No
LNK$LN MOV  R4,R4             Is name length zero?
       JEQ  LNKERR            Yes, error
       CI   R4,7              Is name length > 7?
       JGT  LNKERR            Yes, error
       CLR  @CRULST
       MOV  R4,@SCLEN-1       Store name length for search
       MOV  R4,@SAVLEN        Save device name length
       INC  R4                Adjust it
       A    R4,@SCNAME        Point to position after name
       MOV  @SCNAME,@SAVPAB   Save pointer into device name
```

```
*
*** Search ROM for DSR
*
SROM   LWPI GPLWS           Use GPL workspace to search
       CLR  R1              Version found of DSR etc.
       LI   R12,>0F00       Start over again
NOROM  MOV  R12,R12         Anything to turn off
       JEQ  NOOFF           No
       SBZ  0               Yes, turn it off
NOOFF  AI   R12,>0100       Next ROM'S turn on
       CLR  @CRULST         Clear in case we're finished
       CI   R12,>2000       At the end
       JEQ  NODSR           No more ROMs to turn on
       MOV  R12,@CRULST     Save address of next CRU
       SBO  0               Turn on ROM
       LI   R2,>4000        Start at beginning
       CB   *R2,@HAA        Is it a valid ROM?
       JNE  NOROM           No
       A    @TYPE,R2        Go to first pointer
       JMP SGO2
SGO    MOV  @SADDR,R2       Continue where we left off
       SBO  0               Turn ROM back on
SGO2   MOV  *R2,R2          Is address a zero
       JEQ  NOROM           Yes, no program to look at
       MOV  R2,@SADDR       Remember where we go next
       INCT R2              Go to entry point
       MOV  *R2+,R9         Get entry address
*
*** See if name matches
*
       MOVB @SCLEN,R5       Get length as counter
       JEQ  NAME2           Zero length, don't do match
       CB   R5,*R2+         Does length match?
       JNE  SGO             No
       SRL  R5,8            Move to right place
       LI   R6,NAMBUF       Point to NAMBUF
NAME1  CB   *R6+,*R2+       Is character correct?
       JNE  SGO             No
       DEC  R5              More to look at?
       JNE  NAME1           Yes
NAME2  INC  R1              Next version found
       MOV  R1,@SAVVER      Save version number  !!Could be used to avoid
       MOV  R9,@SAVENT      Save entry address   !!another lookup on
       MOV  R12,@SAVCRU     Save CRU address     !!subsequent calls
       BL   *R9             Match, call subroutine
       JMP  SGO             Not right version
       SBZ  0               Turn off ROM
       LWPI DLNKWS          Select DSRLNK workspace
       MOV  R9,R0           Point to flag byte in PAB
       BLWP @VSBR           Read flag byte
       SRL  R1,13           Just want the error flags
       JNE  IOERR           Error!
       RTWP
*
*** Error handling
*
NODSR  LWPI DLNKWS     Select DSRLNK workspace
LNKERR CLR  R1         Clear the error flags
IOERR  SWPB R1
       MOVB R1,*R13    Store error flags in calling R0
       SOCB @H20,R15   Indicate an error occured
       RTWP            Return to caller
```

```
0001                ************************************************************
0002                * THE FOLLOWING SAMPLE PROGRAM DEMONSTRATES HOW TO OBTAIN  *
0003                * RANDOM NUMBERS FROM ASSEMBLY LANGUAGE.                   *
0004                *                                                          *
0005                * THIS PROGRAM:                                            *
0006                *     - CLEARS THE SCREEN. (LINES 0015-0020)               *
0007                *     - FORMULATES A RANDOM NUMBER. (LINES 0026-0033)      *
0008                *     - FILLS THE SCREEN WITH THE ASCII CHARACTER ASSO-    *
0009                *       CIATED WITH THE RANDOM NUMBER. (LINES 0034-0037)   *
0010                *     - RETURNS TO THE BEGINNING OF THE PROGRAM AND        *
0011                *       REPEATS (LINES 0038-0040)                          *
0012                *                                                          *
0013                * TO RUN:                                                  *
0014                *                                                          *
0015                *     - CHOOSE OPTION #3 (LOAD AND RUN) FROM THE E/A       *
0016                *       SELECTION LIST.                                    *
0017                *     - ENTER THE OBJECT FILE NAME OF THE PROGRAM IN       *
0018                *       RESPONSE TO THE PROMPT 'FILE NAME?'.               *
0019                *     - ENTER 'START' IN RESPONSE TO THE PROMPT            *
0020                *       'PROGRAM NAME?'                                    *
0021                *                                                          *
0022                * TO STOP THE PROGRAM, TURN THE COMPUTER OFF.              *
0023                *                                                          *
0024                ************************************************************
0025                       DEF  START         *
0026                       REF  VSBW          *
0027                *
0028      83E0      GPLWS  EQU  >83E0         * p. 406 E/A MANUAL
0029      837C      STATUS EQU  >837C         * p. 405 E/A MAUNAL
0030      83C0      RAND   EQU  >83C0         * p. 406 E/A MANUAL
0031 0000           MYWSP  BSS  >20           * RESERVE ROOM FOR PROG WORKSPACE
0032                *
0033 0020 02E0      START  LWPI MYWSP         * LOAD PROGRAM WORKSPACE
     0022 0000'
0034                *
0035                * INITIALIZE SCREEN TABLE
0036                *
0037 0024 04C0             CLR  R0            * START ADDRESS OF SCREEN
0038 0026 0201             LI   R1,>2000      * BLANK SPACE
     0028 2000
0039 002A 0420      LOOP   BLWP @VSBW         * p. 248 E/A MANUAL
     002C 0000
0040 002E 0580             INC  R0            * INCREMENT ADDRESS
0041 0030 0280             CI   R0,768        * END OF SCREEN?
     0032 0300
0042 0034 16FA             JNE  LOOP          * NO, DO IT AGAIN
0043                *
0044                * GET RANDOM NUMBER
0045                *
0046 0036 04C0             CLR  R0            * RESET R0 TO BEGINNING OF SCREEN
0047                *
0048 0038 02E0      LOOP1  LWPI GPLWS         * LOAD GPL WORKSPACE
     003A 83E0
0049 003C 0204             LI   R4,28645      * THE NEXT FOUR LINES FORMULATE A
     003E 6FE5
0050 0040 3920             MPY  @RAND,R4      * RANDOM ASCII VALUE IN THE MSB
     0042 83C0
```

```
0044 0225          AI   R5,31417       * OF R5 DEPENDENT UPON THE TWO BYTE
0046 7AB9
0048 C805          MOV  R5,@RAND       * VALUE RETRIEVED FROM >83C0.
004A 83C0
               *                       * SEE THE SOURCE CODE OF TOMBSTONE
               *                       * CITY THAT WAS PACKAGED WITH THE
               *                       * E/A SOFTWARE. (LINES 1310-1317)
               *
004C 02E0          LWPI MYWSP          * RELOAD PROGRAM WORKSPACE
004E 0000'
               *
               * PRINT ASCII CHARACTER TO SCREEN.  REMEMBER, ASCII VALUES
               * 127-255 ARE NOT DEFINED AND WILL APPEAR AS BLANKS WHEN
               * PRINTED TO THE SCREEN.
               *
0050 C060          MOV  @RAND,R1       * MOVE THE WORD AT >83C0 INTO R1
0052 83C0
0054 0420  LOOP2   BLWP @VSBW          * PRINT IT TO SCREEN
0056 002C'
0058 0580          INC  R0             * INCREMENT R0 BY 1
005A 0280          CI   R0,768         * END OF SCREEN?
005C 0300
005E 16FA          JNE  LOOP2          * NO, DO IT AGAIN
               *
0060 04C0          CLR  R0             * START OF SCREEN IMAGE TABLE
0062 0201          LI   R1,>2000       * BLANK SPACE
0064 2000
0066 10E1          JMP  LOOP           * RETURN TO BEGINNING OF PROGRAM
                   END
0000 ERRORS
```